

# Collaborative Filtering with User-Item Co-Autoregressive Models

Chao Du<sup>†</sup> Chongxuan Li<sup>†</sup> Yin Zheng<sup>‡</sup> Jun Zhu<sup>†</sup> Cailiang Liu<sup>‡</sup> Hanning Zhou<sup>‡</sup> Bo Zhang<sup>†</sup>

Dept. of Comp. Sci. & Tech., State Key Lab of Intell. Tech. & Sys., TNList Lab,  
Center for Bio-Inspired Computing Research, Tsinghua University, Beijing, 100084, China<sup>†</sup>  
Hulu LLC., Beijing, 100084<sup>‡</sup>

## ABSTRACT

Besides the success on object recognition, machine translation and system control in games, (deep) neural networks have achieved state-of-the-art results in collaborative filtering (CF) recently. Previous neural approaches for CF are either user-based or item-based, which cannot leverage all relevant information explicitly. We propose CF-UICa, a neural co-autoregressive model for CF tasks, which exploit the structural autoregressiveness in the domains of both users and items. Furthermore, we separate the inherent dependence in this structure under a natural assumption and develop an efficient stochastic learning algorithm to handle large scale datasets. We evaluate CF-UICa on two popular benchmarks: MovieLens 1M and Netflix, and achieve state-of-the-art predictive performance, which demonstrates the effectiveness of CF-UICa.

## CCS Concepts

• **Computing methodologies** → *Machine learning approaches; Machine learning algorithms;*

## Keywords

collaborative filtering; autoregressive model; neural networks

## 1. INTRODUCTION

With the emergence and development of electronic commerce, social networks and music/movie content providers, recommendation techniques are attracting more and more research attentions [7, 37]. Collaborative filtering (CF) [7, 4, 26, 28, 39] is one of the most popular classes of methods for recommendation systems that predict a user's preference for an item based on previous observed preferences in the system.

CF enjoys the benefit of content-independence of the items being recommended. Therefore, it does not need expert knowledge about the items when compared with content-based recommendation [37, 10] and could possibly provide cross-domain recommendations.

The fundamental assumption behind CF is that there exist correlations between preferences observed by recommendation systems, and the correlations can be generalized to unknown preferences. Basically, the correlations can be categorized into two types: *User-User Correlations* (UUCs) describe the correlations between different users' preferences for a same item; and *Item-Item Correlations* (IICs) describe the correlations between preferences for different items expressed by a same user. These two types of underlying cor-

relations usually exist crisscrossing in the partially observed preferences, resulting CF a difficult task.

Over decades, extensive work has studied how to effectively exploit the underlying correlations to make accurate predictions in CF tasks. Early approaches [26, 29, 15] compute the similarities between users or items, namely considering UUCs or IICs, based on the observed preference/rating vectors directly, which perform poorly on the real-life data due to the sparseness of the raw input. To address this problem, latent variable models including Matrix Factorization (MF) and neural network based models have been proposed. MF methods [4, 17, 28, 8, 23] embed both users and items into a shared low dimensional latent space, assuming that the partially observed matrix is low-rank. MF methods take both UUCs and IICs in count implicitly as a prediction is the inner product of the latent vectors of the corresponding user and item. Recently, deep learning achieves state-of-the-art results on image classification [18], speech recognition [11], machine translation [1] and system control in video and board games [25, 32] due to its ability to learn abstract representations hierarchically. Inspired by these advances, neural network based methods [28, 31, 39], which employ highly flexible transformations to learn a compact representation for users and/or items, are widely used as an alternative of MF in CF. These methods focus on employing different neural architectures, including Restricted Boltzmann Machines (RBMs), autoencoders and Neural Autoregressive Distribution Estimators [20] (NADEs) to learn latent features of users or items. Specifically, CF-NADE [39], which extend NADEs to model variable-length vectors, achieves state-of-the-art results on several datasets in CF. Most of neural network methods mentioned only consider either UUCs or IICs explicitly. See more detailed discussion on these related work in Sec. 2.

Though previous neural methods achieved state-of-the-art results in CF, they can be further improved because neither UUCs nor IICs can leverage all of the relevant information completely. To this end, we propose a novel neural autoregressive model for CF named user-item co-autoregressive model (CF-UICa). The name indicates that the model is autoregressive in the domain of both users and items. As a result, on one hand, CF-UICa can make more accurate predictions based on both UUCs and IICs. On the other hand, the structural autoregressiveness along both the users and items makes the model highly coupled inside and hard to train. To tackle the training problem, we separate the inherent dependence in this structure under certain natural assumption. We further develop a stochastic optimiza-

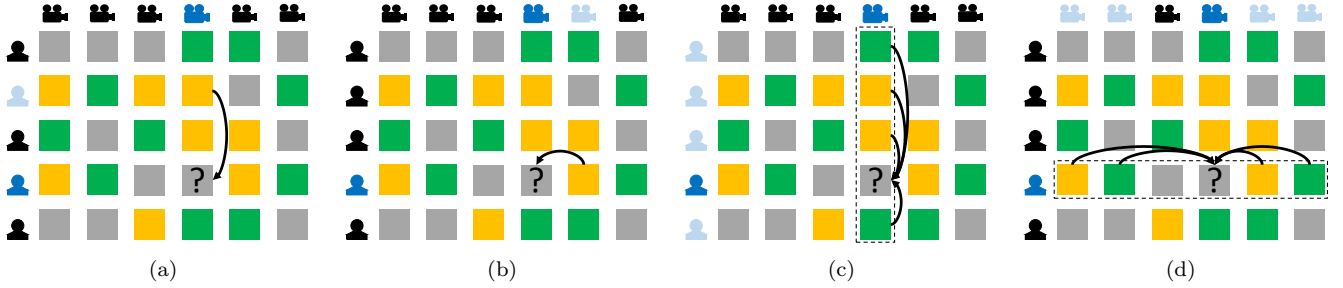


Figure 1: Illustration of predictions in a toy recommendation system with 5 users and 6 items (better viewed in color). Each square colored in green/yellow/gray corresponds to a positive/negative/unobserved preference, respectively. The preference (whose associating user and item are colored in deep blue) being predicted is marked with a question mark. (a) Predicting with a single User-User Correlation: the preference is predicted according to the preference of another user (labeled as light blue) whose behavior (preferences on other items) are related. (b) Predicting with a single Item-Item Correlation. (c) Predicting with multiple User-User Correlations. (d) Predicting with multiple Item-Item Correlations.

tion algorithm for CF-UIcA and make it applicable for large datasets. The model is evaluated on two real world benchmarks: MovieLens 1M and Netflix. We show that our model achieves state-of-the-art results on predictive performance. In addition, visualization results demonstrate that CF-UIcA can learn semantic patterns without extra supervision.

## 2. RELATED WORK

Collaborative filtering methods make predictions based on user behavior. The intrinsic causation of why CF methods work is that a user’s behavior (of expressing preferences to items) could reveal certain kinds of patterns. Namely, a user may have interrelated preferences for a group of items, e.g., comedies, 80’s movies or movies directed by Steven Spielberg. The phenomena involves two types of information: *User-User Correlations* (UUC) and *Item-Item Correlations* (IIC). UUC depicts that a user’s preference for an item is usually related to another user’s preference for the same item, especially when they have similar tastes. An example is shown in Fig. 1a. IIC depicts that a user’s preference for an item is also related to his/her preference for another item, especially when the two items are similar in nature, as shown in Fig. 1b. Predictions are then possible to be made by integrating these correlations. Specifically, one can make prediction by referring to multiple UUCs of the preference being predicted. Fig. 1c shows all the UUCs of the unknown preference marked by the question mark. Intuitively, integrating all the UUCs potentially leads to a more precise prediction than using any one of them. Similarly, Fig. 1d shows all the IICs of the preference being predicted.

Existing CF methods either implicitly or explicitly exploits the above correlations. Early methods model the correlations by similarity functions. One of the most famous method is k-NN collaborative filtering [26], which first computes the top  $k$  nearest users of the active user (whose preference is being predicted) and makes prediction by combining the preferences of the top  $k$  nearest users. Later, [29, 15] proposed item-based k-NN methods. These methods can be viewed as making prediction with the  $k$  UUCs or IICs explicitly. The fact that the raw ratings domain, on which the similarities are computed, is usually very sparse in real-life data leads to poor performance.

Matrix factorization (MF) plays an important role in CF methods. Instead of viewing users and items as vectors in

the ratings domain, MF characterizes both users and items by vectors in a same latent space, whose dimension (rank) is much smaller than the number of users and items. The ratings are modeled with the inner products of the latent vectors of the active users and items. Early work [4, 30] adopt Singular Value Decomposition (SVD) to CF. Probabilistic Matrix Factorization (PMF) [24], extend MF to a probabilistic model by adapting Gaussian distributions over the ratings. Later, a lot of methods are proposed under probabilistic assumption [27, 8, 9, 22, 38]. Recent approaches improve MF by loosening the constraints of linearity and low-rank assumption [17, 23, 6]. MF methods make predictions with the learned latent vectors of the users and the items directly. The correlations are implied in the latent vectors.

Accompany with the success of deep learning methods in many tasks [18, 33, 13], CF methods based on neural networks has been proposed and achieve state-of-the-art results. [28] propose RBM-CF, a CF methods based on RBM, and achieve great success in the Netflix prize challenge [3]. [31] proposed AutoRec, an discriminative model based on autoencoder. Recently, [39] proposed CF-NADE, an tractable model based on NADE [19] and achieve state-of-the-art performance on several CF benchmarks. RBM-CF, AutoRec and CF-NADE are common in that they all build different models for different users with all the models sharing parameters and that they make predictions for a user according to all the ratings from him/her. Note that in these methods, the role of users and items are exchangeable, which results they all have a user-based and an item-based variants. As a result, these methods make prediction with either all the UUCs or all the IICs explicitly.

Our proposed CF-UIcA captures both UUCs and IICs explicitly and simultaneously. Similar with CF-NADE, we adopt neural autoregressive architectures to model the probabilities of ratings. However, CF-NADE models the rating vectors of each user, resulting the users are independent with each other while in order to model UUCs and IICs at the same time, CF-UIcA models the ratings across all users and items, which is the crucial difference between CF-UIcA and CF-NADE. Moreover, since in CF-UIcA the users are not independent with each other, a stochastic learning algorithm is not trivial to derive, which is another contribution of this paper.

### 3. METHOD

In this section we develop a collaborative filtering method which can model both UUCs and IICs with autoregressive architectures.

#### 3.1 Problem Formalization and Notations

Suppose that in a recommendation system there are  $N$  users and  $M$  items. Each user may have expressed preferences to various items by giving them ratings. A rating can be represented by a  $(user, item, rating)$  triple, e.g.  $(i, j, R_{i,j})$  denotes that the user  $i$  gives the item  $j$  a rating  $R_{i,j}$ . Depending on the recommendation system the rating  $R_{i,j}$  may have different forms such as 1- $K$  stars, binary (viewed/not viewed) and ternary (like/dislike/not viewed) scales. In this work we consider the most prevalent case that a rating can take an integer number from 1 to  $K$  ( $K$ -star scale). Suppose in the system we have  $D$  triples  $\{(i_d, j_d, R_{i_d, j_d})\}_{d=1}^D$  observed, where  $D$  is often much smaller than the total number  $N \times M$ , we denote all the  $D$  ratings by  $\mathcal{D} = \{R_{i_d, j_d}\}_{d=1}^D$ , which forms the training set of collaborative filtering methods. The objective of collaborative filtering is then to predict an unknown rating  $R_{i^*, j^*} \notin \mathcal{D}$  based on all known ratings.

It is common to represent the recommendation system with a rating matrix  $\mathbf{R}$  of size  $N \times M$  such that each row corresponds to a user, each column corresponds to an item, and the entry in row  $i$  and column  $j$  corresponds to the rating  $R_{i,j}$ . The training set  $\mathcal{D}$  is then the observed subset of  $\mathbf{R}$ . Note that we also denote the training  $(user, item)$  pairs  $\{(i_d, j_d)\}_{d=1}^D$  as  $\mathcal{D}$ , if no confusion.

Let  $[N] := \{1, \dots, N\}$  denote the set of the integers and  $\sigma$  denote a permutation of  $[N]$ . We denote  $\sigma_i$  and  $\sigma_{<i}$  as the  $i$ -th dimension of  $\sigma$  and the  $i-1$  first dimensions before  $i$ , respectively. We denote the set of all the permutations of  $[N]$  as  $\mathcal{S}_N$ . Following the common convention, we use subscripts for indexing. As already used,  $R_{i,j}$  denotes the entry in row  $i$  and column  $j$  of  $\mathbf{R}$ .  $\mathbf{R}_{i,:}$  and  $\mathbf{R}_{:,j}$  denote the  $i$ -th row and the  $j$ -th column of  $\mathbf{R}$ , respectively. Complex indexing can be inferred easily. For example,  $\mathbf{R}_{\sigma_{<i}, j}$  denotes the first  $i-1$  elements of  $\mathbf{R}_{:,j}$  indexed by  $\sigma$ . 3-dimensional matrices are indexed in a similar way. For any indexing  $\mathcal{I}$ , we define  $\mathcal{D}_{\mathcal{I}} = \mathbf{R}_{\mathcal{I}} \cap \mathcal{D}$ .

#### 3.2 The Model

We start with a very generic autoregressive assumption to model the probability of the whole rating matrix:

$$p(\mathbf{R}) = \prod_{t=1}^{N \times M} p(R_{o_t} | R_{o_{<t}}), \quad (1)$$

where  $o$  is a permutation of the Cartesian product  $[N] \times [M] = \{(i, j) | i \in [N], j \in [M]\}$  that serves as an ordering of all the entries in the rating matrix  $\mathbf{R}$ , and  $R_{o_{<t}}$  denotes the first  $t-1$  entries of  $\mathbf{R}$  indexed by  $o$ . For example, if  $o_t = (i', j')$  then  $R_{o_t} = R_{i', j'}$ . Let  $o_t^i = i'$  denotes the first dimension of  $o_t$  which indexes the row of  $\mathbf{R}$  and  $o_t^j = j'$  denotes the dimension of  $o_t$  which indexes the column. Basically, there are  $(N \times M)!$  possible orderings of all the entries in  $\mathbf{R}$ . For now we assume that  $o$  is fixed. If we consider  $o$  as the ordering of timestamps of ratings received by the system, then the conditional in (1) means the model predicts rating  $R_{o_t}$  at time  $t$  depends on all existing (known) ratings.

According to Sec. 2, both UUCs and IICs will be informative for predicting. We define a conditional model that exploits both UUCs and IICs:

$$p(R_{o_t} | R_{o_{<t}}) = p(R_{o_t} | R_{o_{<t}|i=o_t^i}, R_{o_{<t}|j=o_t^j}), \quad (2)$$

where we have defined  $R_{o_{<t}|i=o_t^i} = \{R_{o_{t'}} : t' < t, o_{t'}^i = o_t^i\}$  as the ratings from user  $o_t^i$  in  $R_{o_{<t}}$ , which forms all the IICs of  $R_{o_t}$  (by time  $t$ ) and  $R_{o_{<t}|j=o_t^j} = \{R_{o_{t'}} : t' < t, o_{t'}^j = o_t^j\}$  is the ratings for item  $o_t^j$  in  $R_{o_{<t}}$ , which forms all the UUCs of  $R_{o_t}$  (by time  $t$ ). Fig. 2 (Left) illustrates the two parts of the condition.

##### 3.2.1 The Conditional Model

A naive approach to model (2) can be simply adopting a weighted sum of the scores in  $R_{o_{<t}|i=o_t^i}$  and  $R_{o_{<t}|j=o_t^j}$ . However, this linear model may be not flexible enough. Inspired by NADE [20] and CF-NADE [39], we model the conditionals in (2) with neural networks due to the rich expressive ability.

Specifically, CF-UICa models the UUCs and IICs of  $R_{o_t}$  with hidden representations respectively:

$$\mathbf{h}^U(R_{o_{<t}|j=o_t^j}) = f\left(\sum_{t' < t: o_{t'}^j = o_t^j} \mathbf{W}_{:, o_{t'}^j, R_{o_{t'}}}^U + \mathbf{c}^U\right), \quad (3)$$

$$\mathbf{h}^I(R_{o_{<t}|i=o_t^i}) = f\left(\sum_{t' < t: o_{t'}^i = o_t^i} \mathbf{W}_{:, o_{t'}^i, R_{o_{t'}}}^I + \mathbf{c}^I\right), \quad (4)$$

where  $f(\cdot)$  is the activation function, such as  $\tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$ ,  $\mathbf{W}^U \in \mathbb{R}^{H_U \times N \times K}$  and  $\mathbf{W}^I \in \mathbb{R}^{H_I \times M \times K}$  are 3-dimensional weight matrices,  $\mathbf{c}^U \in \mathbb{R}^{H_U}$  and  $\mathbf{c}^I \in \mathbb{R}^{H_I}$  are the bias terms.  $H_U$  and  $H_I$  are the dimensions of the hidden representation for UUCs and IICs, respectively. The column  $\mathbf{W}_{:, j, k}^I$  denotes how much “rating item  $j$  with  $k$ ” contributes to the hidden representation of IICs while the column  $\mathbf{W}_{:, i, k}^U$  is the contribution of “user  $i$  gives a rating  $k$ ” to the hidden representation of UUCs. Fig. 2 illustrates how  $\mathbf{h}^U$  and  $\mathbf{h}^I$  are computed for the case where  $K = 2$ .

CF-UICa explains the hidden representations of UUCs and IICs for  $R_{o_t}$  by computing the activations:

$$s_{o_t, k}^U(R_{o_{<t}|j=o_t^j}) = \mathbf{b}_{o_t^j, k}^U + \mathbf{V}_{:, o_t^j, k}^U \top \mathbf{h}^U(R_{o_{<t}|j=o_t^j}), \quad (5)$$

$$s_{o_t, k}^I(R_{o_{<t}|i=o_t^i}) = \mathbf{b}_{o_t^i, k}^I + \mathbf{V}_{:, o_t^i, k}^I \top \mathbf{h}^I(R_{o_{<t}|i=o_t^i}), \quad (6)$$

where  $\mathbf{V}^U \in \mathbb{R}^{H_U \times N \times K}$  and  $\mathbf{V}^I \in \mathbb{R}^{H_I \times M \times K}$  are 3-dimensional weight matrices,  $\mathbf{b}^U \in \mathbb{R}^{N \times K}$  and  $\mathbf{b}^I \in \mathbb{R}^{M \times K}$  are the bias terms. The column  $\mathbf{V}_{:, j, k}^I$  is the coefficients that determines how the hidden representation of IICs affects the activation  $s_{j, k}^I$  for item  $j$  with rating  $k$ . Higher activation  $s_{j, k}^I$  indicates that the considered IICs suggest higher probability that the user will rate item  $j$  with rating  $k$ . Fig. 2 illustrates how  $s^U$  and  $s^I$  are computed.

To combine the activations of UUCs and IICs of  $R_{o_t}$  and produce a probability distribution, the final conditional model is defined by a softmax function on the summation of two types of activations:

$$p(R_{o_t} = k | R_{o_{<t}}) = \frac{\exp\left(s_{o_t, k}^U(R_{o_{<t}|i=o_t^i}) + s_{o_t, k}^I(R_{o_{<t}|j=o_t^j})\right)}{\sum_{k'=1}^K \exp\left(s_{o_t, k'}^U(R_{o_{<t}|i=o_t^i}) + s_{o_t, k'}^I(R_{o_{<t}|j=o_t^j})\right)}. \quad (7)$$

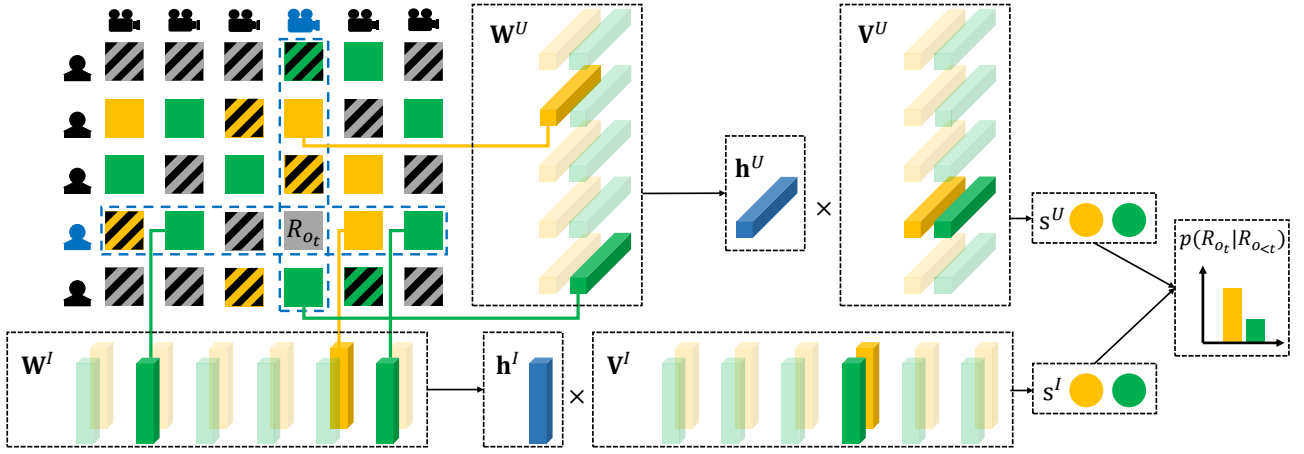


Figure 2: An illustration of the proposed model (better viewed in color). The yellow, green and gray entries are interpreted same as in Fig. 1. Suppose  $R_{o_t}$  is the current rating being modeled. The black striped line marks the entries of  $R_{o_{>t}}$ . The blue dashed boxes line out the UUCs and IICs for  $R_{o_{>t}}$ . The cuboids represent the columns of  $\mathbf{W}^U$ ,  $\mathbf{W}^I$ ,  $\mathbf{V}^U$ ,  $\mathbf{V}^I$ , with the color corresponding to the rating. The hidden representations  $\mathbf{h}^U$  and  $\mathbf{h}^I$  are computed as the summation over the corresponding columns (with uncorresponding columns are marked with lighter colors) of UUCs and IICs. The bias terms are omitted for clarity.

Fig. 2 illustrates the computation flow of the conditional model.

### 3.2.2 Models with Different Orderings

We now give the objective function for CF-UICa. So far the model is defined under a fixed ordering  $o$ . As stated before there are  $(N \times M)!$  possible orderings, we thus need to handle models with different orderings. We are inspired by the idea of EoNADE [35] of sharing parameters across all models with different orderings. However, the problem is quite different due to the autoregressiveness is much more structural under the collaborative filtering setting.

Inference of an unknown rating  $R_{i^*,j^*} = R_{o_{t^*}}$  under the model defined above can be formalized as  $p(R_{i^*,j^*} | R_{\text{cond}})$ , where  $R_{\text{cond}}$  is a subset of elements in  $\mathbf{R}$  to condition on. The inference of  $R_{i^*,j^*}$  is easy as long as  $R_{\text{cond}}$  is exactly the first  $t^* - 1$  elements of  $\mathbf{R}$  indexed by the ordering  $o$ . Under such a case the inference can be done by simply applying the conditional model defined in (2). Otherwise, we have to do marginalisation or posterior inference, which will be very hard. In order to simply utilize the conditional model (2) to deal with any inference task, we need to train as many models as there are possible orderings. A naive approach is to train  $(N \times M)!$  different models to cover all possible ordering  $o$ , which is, obviously, not viable due to its factorial complexity. For collaborative filtering, we are only interested in the inference tasks whose  $R_{\text{cond}} = \mathcal{D}$  and thus the orderings indexed by which the  $\mathcal{D}$  are always the first  $D$  elements of  $\mathbf{R}$ . However, there are still  $D!(N \times M - D)!$  different orderings satisfying this constraint.

Inspired by EoNADE [36], we consider a parameter tying strategy that simply uses the same weight matrices and bias parameters across the models with different orderings. Specifically, among models with different orderings, the parameters  $\mathbf{W}^U$ ,  $\mathbf{W}^I$ ,  $\mathbf{c}^U$ ,  $\mathbf{c}^I$  in the conditional model (3), (4) are shared for any ordering  $o$ . And similarly the parameters  $\mathbf{V}^U$ ,  $\mathbf{V}^I$ ,  $\mathbf{b}^U$ ,  $\mathbf{b}^I$  are shared in (6),(5). We thus can rewrite the likelihood of the rating matrix as  $p(\mathbf{R} | \boldsymbol{\theta}, o)$

where we define  $\boldsymbol{\theta} = \{\mathbf{W}^A, \mathbf{V}^A, \mathbf{c}^A, \mathbf{b}^A\}_{A=U,I}$  the model parameters and view the ordering  $o$  as a random variable. As analyzed above, the interested orderings are those placing  $\mathcal{D}$  at the beginning. We denote the set of these orderings as  $\mathfrak{S}_D \subseteq \mathfrak{S}_{(N \times M - D)}$ .

The objective is then the expected (over all interested ordering  $o$ ) negative log-likelihood of the training set:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{o \in \mathfrak{S}_D} -\log p(\mathcal{D} | \boldsymbol{\theta}, o) \quad (8)$$

$$= -\mathbb{E}_{o \leq D \in \mathfrak{S}_D} \sum_{d=1}^D \log p(R_{o_d} | R_{o_{<d}}, \boldsymbol{\theta}, o), \quad (9)$$

Where  $\mathfrak{S}_D$  serves as the set of permutations of  $\mathcal{D}$ . The expectation over the ordering of  $\mathbf{R} \setminus \mathcal{D}$  is omitted since the log-likelihood of  $\mathcal{D}$ , which is the first  $D$  elements of  $\mathbf{R}$  indexed by  $o$ , is independent with  $o_{>D}$ . From now we will use the notation  $o$  instead of  $o_{\leq D}$  for simplicity.

At the test stage, CF-UICa predict  $R_{i^*,j^*}$  by computing the expected rating under the the distribution conditioned on the training set:

$$\hat{R}_{i^*,j^*} = \mathbb{E}_{p(R_{i^*,j^*}=k|\mathcal{D})}[k], \quad (10)$$

where the probability  $p(R_{i^*,j^*} = k | \mathcal{D})$  is computed according to (7), with  $\mathbf{h}^I$ ,  $s^I$  computed using  $\mathcal{D}_{i^*,:}$  which denotes all the training ratings from user  $i^*$ :

$$\mathbf{h}^I(\mathcal{D}_{i^*,:}) = f\left(\sum_{(i,j) \in \mathcal{D}, i=i^*} \mathbf{W}_{:,j,R_{i,j}}^I + \mathbf{c}^I\right), \quad (11)$$

$$s_k^I(\mathcal{D}_{i^*,:}) = b_{j^*,k}^I + \mathbf{V}_{:,j^*,k}^I \mathbf{h}^I(\mathcal{D}_{i^*,:}), \quad (12)$$

$\mathbf{h}^U$ ,  $s^U$  computed using  $\mathcal{D}_{:,j^*}$  which denotes all the training ratings of item  $j^*$  in a similar way.

## 3.3 Learning

The remaining challenge is to optimize the objective (9). Since it involves an expectation over factorially many orderings and each term involves estimating the log-likelihood of the entire training set, it is impractical to play with (9)

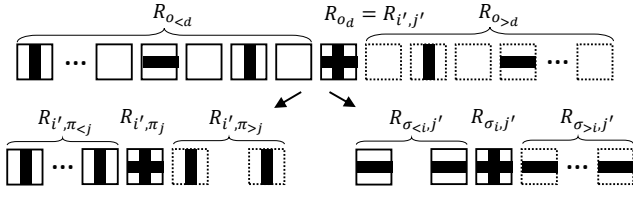


Figure 3: Illustration of the approximation of the randomness of  $\mathbf{R}_{i',-j'} \cap R_{o<d}$  and  $\mathbf{R}_{-i',j'} \cap R_{o<d}$ . The squares represent entries of  $\mathbf{R}$ . The entries of  $\mathbf{R}_{i',-j'}$  are marked with horizontal black bars and the entries of  $\mathbf{R}_{-i',j'}$  are marked with vertical black bars. (Top) The procedure of determining  $\mathbf{R}_{i',-j'} \cap R_{o<d}$  and  $\mathbf{R}_{-i',j'} \cap R_{o<d}$  is equivalently by first permuting all the entries of  $\mathbf{R}$  according to the random ordering  $o$  from left to right and then selecting all the entries with horizontal bars and vertical bars, respectively, from the entries on the left of  $R_{o_d}$ . (Bottom) This procedure is approximated by first permuting  $\mathbf{R}_{i',-j'}$  and  $\mathbf{R}_{-i',j'}$  according to two independent orderings  $\sigma$  and  $\pi$  and then selecting the entries on the left of  $R_{i',\pi_j}$  as  $\mathbf{R}_{i',-j'} \cap R_{o<d}$  and selecting the entries on the left of  $R_{\sigma_i,j'}$  as  $\mathbf{R}_{-i',j'} \cap R_{o<d}$ .

directly. A common strategy to deal with complicate integration or large-scale data is to adopt stochastic optimization approaches, e.g. stochastic gradient descent (SGD) [5, 16], which require an (approximately) unbiased estimator of the objective or its gradient and is guaranteed to (approximately) converge. SGD and its variants have been widely adopted in various areas due to its efficiency for large scale data. It is also adopted by many CF methods [31, 39]. In the following we derive an approximation of (9) and provide a complete algorithm for training the model.

We first show a simple identity which is useful for later derivation.

**PROPOSITION 1.** *Let  $\Omega = \{\omega_t\}_{t=1}^T$  be a set of  $T$  elements. Let  $o$  be a random permutation (ordering) of  $\Omega$  that is uniformly distributed over all possible permutations of  $\Omega$ . Let  $g(x, o)$  be some function defined on  $x$  and  $o$  where  $x$  is an arbitrary variable. The following equations hold:*

$$\sum_{i=1}^T \mathbb{E}_o g(i, o) = \sum_{i=1}^T \mathbb{E}_{o_i} \mathbb{E}_{o|o_i} g(i, o) = \sum_{\omega \in \Omega} \mathbb{E}_i \mathbb{E}_{o|o_i=\omega} g(i, o), \quad (13)$$

where  $\mathbb{E}_{o|o_i}$  is taking the expectation over  $o$  with  $o_i$  given.

The key point of the proposition is that the ordering can be split into two parts:  $o_i$  is the  $i$ -th dimension of  $o$  and  $o|o_i$  the other dimensions of  $o$  except  $o_i$ . The expectation over  $o_i$  can be calculated by averaging over all the elements in  $\Omega$ .

By exchanging the order of the expectation and the summation in (9) and adopting the proposition, we get:

$$\mathcal{L}(\theta) = - \sum_{(i',j') \in \mathcal{D}} \mathbb{E}_{o|o_d=(i',j')} \log p(R_{i',j'} | R_{o<d}, \theta, o), \quad (14)$$

where we have assumed each ordering has equal probability.

According to the definition of the conditional model from (3) to (7), The log-probability of  $R_{i',j'}$  in (14) depends on at most  $\mathbf{R}_{i',-j'} = \mathbf{R}_{i',-j'} \setminus \{R_{i',j'}\}$  and  $\mathbf{R}_{-i',j'} = \mathbf{R}_{-i',j'} \setminus \{R_{i',j'}\}$  despite of  $o$ . Specifically, given  $o_d = (i', j')$ , the log-probability of  $R_{i',j'}$  depends on  $\mathbf{R}_{i',-j'} \cap R_{o<d}$  and  $\mathbf{R}_{-i',j'} \cap R_{o<d}$ . Theoretically, these two set (as two functions of the random variable  $d$  and the random variables  $o$  given  $o_d = (i', j')$ )

are not independent. We make a simple approximation that  $\mathbf{R}_{i',-j'} \cap R_{o<d}$  and  $\mathbf{R}_{-i',j'} \cap R_{o<d}$  are determined independently by the random orderings of  $\mathbf{R}_{i',-j'}$  and  $\mathbf{R}_{-i',j'}$ , respectively, instead of by the ordering of  $\mathbf{R}$ . Fig. 3 illustrates the approximation. Breaking the expectation  $\mathbb{E}_d \mathbb{E}_{o|o_d=(i',j')}$  into two parts according to the above leads to an approximation of  $\mathcal{L}(\theta)$ :

$$\mathcal{L}(\theta) \approx \tilde{\mathcal{L}}(\theta) = - \sum_{i'=1}^N \sum_{j'=1}^M \mathbb{E}_i \mathbb{E}_{\sigma|i=i'} \mathbb{E}_j \mathbb{E}_{\pi|\pi=j'} \quad (15)$$

$$\log p(R_{i',j'} | \mathcal{D}_{i',\pi_{<j}}, \mathcal{D}_{\sigma_{<i},j'}, \theta, \sigma, \pi) \mathbb{I}_{[(i',j') \in \mathcal{D}]},$$

where  $\sigma$  is a random permutation of  $[N]$  that serves as an ordering of all the users and  $\pi$  a random permutation of  $[M]$  that serves as an ordering of all the items,  $\mathcal{D}_{i',\pi_{<j}}$  and  $\mathcal{D}_{\sigma_{<i},j'}$  are the approximation of  $\mathbf{R}_{i',-j'} \cap R_{o<d}$  and  $\mathbf{R}_{-i',j'} \cap R_{o<d}$ , respectively. Note the summation over  $\mathcal{D}$  is replaced by an equivalent representation using an indicator function  $\mathbb{I}_{[(i',j') \in \mathcal{D}]}$ . Adopting the proposition again leads to:

$$\tilde{\mathcal{L}}(\theta) = - \sum_{i=1}^N \sum_{j=1}^M \mathbb{E}_{\sigma} \mathbb{E}_{\pi} \log p(R_{\sigma_i,\pi_j} | \mathcal{D}_{\sigma_i,\pi_{<j}}, \mathcal{D}_{\sigma_{<i},\pi_j}, \theta, \sigma, \pi) \mathbb{I}_{[(\sigma_i,\pi_j) \in \mathcal{D}]}, \quad (16)$$

Given  $\sigma$  and  $\pi$ , the log-probability term and the indicator term can be computed easily. From now we omit them for simplicity. The order  $\sigma$  can be split into 3 parts:  $\sigma_{<i}$  the  $i-1$  dimensions at the beginning of  $\sigma$  and  $\sigma_{i:\hat{i}}$  the  $i$ -th to  $(\hat{i}-1)$ -th dimensions in  $\sigma$  and  $\sigma_{\geq \hat{i}}$  the remaining dimensions, for some  $\hat{i} : i < \hat{i} \leq N$ . Similarly, the order  $\pi$  can be split into 3 part  $\pi_{<j}$ ,  $\pi_{j:\hat{j}}$  and  $\pi_{\geq \hat{j}}$  for some  $\hat{j} : j < \hat{j} \leq M$ . Since the log-probability term is independent with  $\sigma_{\geq \hat{i}}$  and  $\pi_{\geq \hat{j}}$ , we have

$$\sum_{i=1}^N \sum_{j=1}^M \mathbb{E}_{\sigma} \mathbb{E}_{\pi} = \sum_{i=1}^N \sum_{j=1}^M \mathbb{E}_{\sigma_{i:\hat{i}}} \mathbb{E}_{\sigma_{<i}} \mathbb{E}_{\pi_{j:\hat{j}}} \mathbb{E}_{\pi_{<j}}, \quad (17)$$

where we omit the log-probability term and the indicator term. Moreover, the log-probability term depends on only  $\sigma_i$  in  $\sigma_{i:\hat{i}}$  and  $\pi_j$  in  $\pi_{j:\hat{j}}$ , the above equation can be rewritten as:

$$\sum_{i=1}^N \sum_{j=1}^M \mathbb{E}_{\sigma} \mathbb{E}_{\pi} = \sum_{i=1}^N \sum_{j=1}^M \mathbb{E}_{\sigma_{i:\hat{i}}} \mathbb{E}_{\pi_{j:\hat{j}}} \mathbb{E}_{\sigma_i} \mathbb{E}_{\pi_j} \mathbb{E}_{\sigma_{<i}} \mathbb{E}_{\pi_{<j}}, \quad (18)$$

where  $\mathcal{S}_{i:\hat{i}}^{\sigma}$  denotes the set of the  $i$ -th to  $(\hat{i}-1)$ -th dimensions of  $\sigma$  and  $\mathcal{S}_{j:\hat{j}}^{\pi}$  denotes the set of the  $j$ -th to  $(\hat{j}-1)$ -th dimensions of  $\pi$ . Finally, the expectation over  $\sigma_i$  and  $\pi_j$  can be calculated by averaging all possible value:

$$\sum_{i=1}^N \sum_{j=1}^M \mathbb{E}_{\sigma} \mathbb{E}_{\pi} = \sum_{i=1}^N \sum_{j=1}^M \mathbb{E}_{\sigma_{i:\hat{i}}} \mathbb{E}_{\pi_{j:\hat{j}}} \frac{1}{|\mathcal{S}_{i:\hat{i}}^{\sigma}|} \sum_{\sigma_i \in \mathcal{S}_{i:\hat{i}}^{\sigma}} \frac{1}{|\mathcal{S}_{j:\hat{j}}^{\pi}|} \sum_{\pi_j \in \mathcal{S}_{j:\hat{j}}^{\pi}} \mathbb{E}_{\sigma_{<i}} \mathbb{E}_{\pi_{<j}}, \quad (19)$$

By sampling  $i, j, \mathcal{S}_{i:\hat{i}}^{\sigma}, \mathcal{S}_{j:\hat{j}}^{\pi}, \sigma_{<i}$  and  $\pi_{<j}$ , We reach an unbiased estimator of (15):

$$\widehat{\mathcal{L}}(\theta) = \frac{N}{\hat{i}-i} \frac{M}{\hat{j}-j} \sum_{\sigma_i \in \mathcal{S}_{i:\hat{i}}^{\sigma}} \sum_{\pi_j \in \mathcal{S}_{j:\hat{j}}^{\pi}} \quad (20)$$

$$\log p(R_{\sigma_i,\pi_j} | \mathcal{D}_{\sigma_i,\pi_{<j}}, \mathcal{D}_{\sigma_{<i},\pi_j}, \theta, \sigma, \pi) \mathbb{I}_{[(\sigma_i,\pi_j) \in \mathcal{D}]}.$$

The gradient  $\nabla_{\theta} \widehat{\mathcal{L}}(\theta)$  can be then combined with SGD algorithms. Note that there are two free parameters  $\hat{i}$  and

**Algorithm 1** CF-UIcA

---

```

1: repeat
2:   Sample  $i \sim [N]$  uniformly
3:   Sample  $j \sim [M]$  uniformly
4:   Select  $\hat{i} \leftarrow \min(i + B_U, N)$ 
5:   Select  $\hat{j} \leftarrow \min(j + B_I, M)$ 
6:   Sample  $\mathcal{S}_{i:\hat{i}}^\sigma \subset [N]$ 
7:   Sample  $\mathcal{S}_{j:\hat{j}}^\pi \subset [M]$ 
8:   for each  $(\sigma_i, \pi_j) \in \mathcal{S}_{i:\hat{i}}^\sigma \times \mathcal{S}_{j:\hat{j}}^\pi$  do
9:     if  $(\sigma_i, \pi_j) \in \mathcal{D}$  then
10:      Sample  $\mathcal{S}_{<i}^\sigma \subset [N] \setminus \mathcal{S}_{i:\hat{i}}^\sigma$ 
11:      Sample  $\mathcal{S}_{<j}^\pi \subset [M] \setminus \mathcal{S}_{j:\hat{j}}^\pi$ 
12:      Evaluate  $\nabla_{\theta} \log p(R_{\sigma_i, \pi_j})$ 
13:   Gradient descent with the mean of the gradients
14: until Converged

```

---

$\hat{j}$ . After  $i$  and  $j$  are sampled, by setting  $\hat{i} - i = B_U$  and  $\hat{j} - j = B_I$  we can control the size of  $\mathcal{S}_{i:\hat{i}}^\sigma$  and  $\mathcal{S}_{j:\hat{j}}^\pi$ . Note that since  $i$  or  $j$  may be sampled very close to  $N$  or  $M$ , we can not always found such  $\hat{i}$  or  $\hat{j}$ . Thus,  $B_U$  and  $B_I$  can then be viewed as the maximum minibatch size of users and items, respectively. The final algorithm is described in Algorithm 1.

### 3.4 Dealing with Large Data

The model described above has one disadvantage that the number of model parameters is linear w.r.t. the number of users and items. For real-life recommendation systems, the number of users is typically of millions or larger, which leads to a too large number of parameters. A common observation of recommendation systems is the redundancy in both users and items that they can usually be divisible into groups with similar preference profiles [7].

Based on the redundancy, we propose to address this problem by clustering the users into groups, with the inside-group users sharing parameters. Specifically, suppose  $z : [N] \mapsto [N']$  maps user  $i \in [N]$  to its cluster  $z(i) \in [N']$ , where  $N'$  is the number of clusters. We make the constraint that

$$\begin{aligned} \mathbf{W}_{:,k}^U &= \mathbf{W}_{:,z(i),k}^U & \mathbf{V}_{:,i,k}^U &= \mathbf{V}_{:,z(i),k}^U \\ \mathbf{c}_i^U &= \mathbf{c}_{z(i)}^U & \mathbf{b}_{i,k}^U &= \mathbf{b}_{z(i),k}^U. \end{aligned} \quad (21)$$

The free dimensions of  $\mathbf{W}^U$  and  $\mathbf{V}^U$  are then reduced to  $H_U \times N' \times K$ , where we can adjust  $N'$  as a trade-off between the conciseness and the capacity of the model.

## 4. EXPERIMENTS

In this section, we present a series of experimental results of the proposed CF-UIcA on two representative recommendation systems datasets: MovieLens 1M [12] and Netflix [3]. MovieLens 1M consist of 1,000,209 ratings of 3,952 movies (items) rated by 6,040 users. Netflix consist of 100,480,507 ratings of 17,770 movies rated by 480,189 users. The ratings in both datasets are 1-5 stars scale, i.e.,  $K = 5$ . Following LLORMA [23], AutoRec [31] and CF-NADE [39], we randomly select 90% of the ratings in each of the datasets as the training set, leaving the remaining 10% of the ratings as the test set. Among the ratings in the training set, 5% are hold out for validation. We compare the predictive performance with state-of-the-art methods in terms of Root Mean

Squared Error (RMSE):

$$RMSE = \sqrt{\frac{\sum_{(i,j) \in \mathcal{D}_{\text{test}}} (\hat{R}_{i,j} - R_{i,j})^2}{D_{\text{test}}}}, \quad (22)$$

where  $R_{i,j}$  is the true rating and  $\hat{R}_{i,j}$  is the rating predicted with (10),  $\mathcal{D}_{\text{test}}$  is the test set of  $D_{\text{test}}$  unknown ratings.

In Sec. 4.1 we describe the experimental settings. In Sec. 4.2 and Sec. 4.3, We compare the RMSE results with state-of-the-art methods on MovieLens 1M and Netflix. We also analyze the sensitivity to key hyperparameters of CF-UIcA and visualize the learned model. Finally, the time and memory complexity are analyzed in Sec. 4.4.

### 4.1 Experimental Settings

In our implementation, we train the model with the difference parameters  $\theta' = \{\mathbf{W}'^A, \mathbf{V}'^A, \mathbf{c}^A, \mathbf{b}'^A\}_{A=U,I}$  instead of  $\theta$ , where  $\mathbf{W}_{:,k}^A = \sum_{y=1}^k \mathbf{W}_{:,y}^A$ ,  $\mathbf{V}_{:,k}^A = \sum_{y=1}^k \mathbf{V}_{:,y}^A$  and  $\mathbf{b}_{:,k}^A = \sum_{y=1}^k \mathbf{b}_{:,y}^A$ . Note that training the model with  $\theta$  and with  $\theta'$  is equivalent theoretically. This is first proposed in [39] and is explained as a regularization. We follow [39] to train our model with difference parameters and observe it works better than training with the origin parameters  $\theta$  for our proposed CF-UIcA. We train our models with  $\theta'$  throughout our experiments.

We also adopt the ordinal cost proposed by [39], which suggests computing the conditional  $p(R_{o_t} = k | R_{o_{<t}})$  as:

$$p(R_{o_t} = k | R_{o_{<t}}) = \prod_{y=1}^k \frac{\exp(s_{o_t,y}^U + s_{o_t,y}^I)}{\sum_{y'=1}^y \exp(s_{o_t,y'}^U + s_{o_t,y'}^I)} \prod_{y=k}^K \frac{\exp(s_{o_t,y}^U + s_{o_t,y}^I)}{\sum_{y'=y}^K \exp(s_{o_t,y'}^U + s_{o_t,y'}^I)}, \quad (23)$$

instead of (7), where  $s_{o_t,y}^U$  is a shorthand for  $s_{o_t,y}^U(R_{o_{<t}|j=o_t^j})$  and  $s_{o_t,y}^I$  is a shorthand for  $s_{o_t,y}^I(R_{o_{<t}|i=o_t^i})$ . [39] has shown computing the conditional with (23) performs better than (7) for CF-NADE as (23) can capture the ordinal nature of user's preference [34]. In our experiments, we observe similar improvement as in [39]. Thus, we compute the conditionals using (23) instead of (7) throughout our experiments.

We use Adam [16] to optimize the objective. The learning rate is set among  $\{0.001, 0.0005, 0.0002\}$ , from which we report the experiment with best performance. Other parameters of Adam is fixed with  $b_1 = 0.1, b_2 = 0.001, \epsilon = 10^{-4}$ . We use weight decay on weight matrices to prevent the model from overfitting. The weight decay term, accompany with the hidden dimensions  $H^U, H^I$  and the minibatch sizes  $B_U, B_V$ , will be explained specifically for each experiments Later.

### 4.2 Experiments on MovieLens 1M

In this section we provide experiments on MovieLens 1M. The minibatch sizes of both users and items are set to 1,000 and we fix the weight decay to 0.0001 throughout the experiments of MovieLens 1M.

#### 4.2.1 Predictive Performance

Since our proposed CF-UIcA has a strong connection with CF-NADE [39] as mentioned in Sec. 2 and Sec. 4.1, we first present a comparison between CF-UIcA and CF-NADE in Fig. 4. CF-NADE models each user with a latent representation, similar with our hidden representation of UUCs or

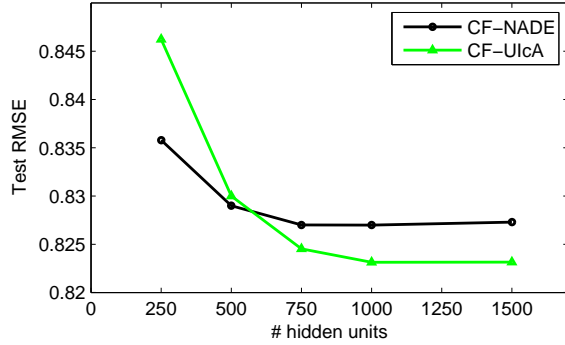


Figure 4: The performance on MovieLens 1M of CF-UIcA and CF-NADE w.r.t. the number of hidden units.

Table 1: Test RMSE of various methods on MovieLens 1M. All the results are taken from [39].

Method	Test RMSE
PMF	0.883
U-RBM	0.881
I-AutoRec	0.874
LLORMA-Local	0.865
I-RBM	0.854
BiasMF	0.845
U-CF-NADE-S (2 layers)	0.845
NNMF	0.843
LLORMA-Local	0.833
I-AutoRec	0.831
I-CF-NADE-S (2 layers)	0.829
CF-UIcA ( $H^U = H^I = 500$ )	<b>0.823</b>

IICs. For fairness, we compare the two methods with the same number of hidden units, where in our CF-UIcA the number of hidden units is  $H^U + H^I$ . Note that in CF-UIcA  $H^U$  is not necessarily equal to  $H^I$ , we choose  $H^U = H^I$  for simplicity. We report the item-based CF-NADE results under best setting as described in [39].

From Fig. 4 we observe that for small number of hidden units, e.g. 250, our method gives a worse result than CF-NADE. This is attributed to the hidden dimensions allocated to the hidden representation of UUCs and IICs are too small ( $H^U = H^I = 125$ ) to capture the underlying information. As the number of hidden units increases, we observe CF-UIcA outperforms CF-NADE since CF-UIcA can capture both UUCs and IICs while the item-based CF-NADE can only capture UUCs. One thing worth mentioning is that the total number of parameters in CF-UIcA is only around 83% of the number of parameters in CF-NADE for MovieLens 1M, when the number of hidden units are same, which implies CF-UIcA can capture more information than CF-NADE while using fewer parameters.

Table 1 compares the performance of CF-UIcA with state-of-the-art methods on MovieLens 1M. The hidden dimensions are  $H^U = H^I = 500$ . Our method achieves an RMSE of 0.823, which outperforms the existing strong baselines.

#### 4.2.2 Visualization

The MovieLens 1M dataset comes with side informations. Specifically, each of the 3,952 movies are marked with one or

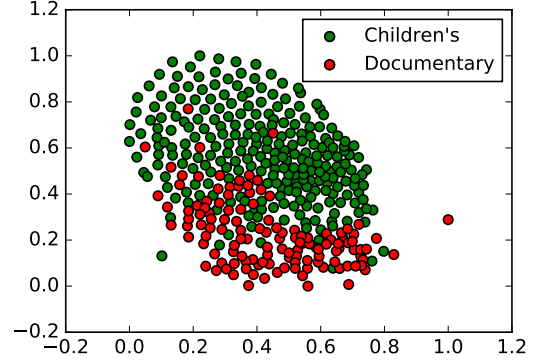


Figure 5: t-SNE embedding of the learned vectors for MovieLens 1M.

more genres. There are totally 18 different genres includes *Action*, *Children's*, *Drama* and etc.. We show our learned model by visualizing the learned weight matrices  $\mathbf{V}^I$ . Specifically, once the model is learned,  $\mathbf{V}_{:,j,:}^I$  can be viewed as  $H^I \times K$  dimensional vectors associated with item  $j$ . We apply t-SNE [21] to embed these vectors into a 2-dimensional plane. Since most movies are labeled with multiple genres which results the genres are not exclusive, we show the t-SNE embedding of two most exclusive genres: *Children's* and *Documentary*. Fig. 5 shows the result of embedding. We can observe the learned vectors are distributed semantically in the gross.

### 4.3 Experiments on Netflix

In this section we provide experiments on Netflix. The Netflix dataset is much bigger than MovieLens 1M, especially the number of users. We opt to use the method described in Sec. 3.4. We cluster all the 480,189 users into 10K, 15K and 20K groups, respectively. To cluster the users, we first run matrix factorization [14] with rank 100 on the training set. (Predicting the test set with the learned vectors by MF gives an RMSE of 0.865.) Then the clustering process is simply done by running a K-means clustering algorithm on the learned latent vectors of all users by MF. For CF-UIcA, the weight decay is set to  $5 \times 10^{-6}$  as the dataset is so big that a strong weight decay is not needed. The minibatch size of users and items are set to  $B_U = 4,000$  and  $B_I = 1,000$ . The hidden dimension are  $H^U = H^I = 500$ .

Fig. 6 shows the performance of CF-UIcA with different number of clusters. We observe that the performance improves as the number of clusters of users increases, which can be attributed to using more clusters increases the ability of capturing the variety among users. Another observation is that the performance can be potentially improved by further increasing the number of clusters.

Table 2 summarizes our best results and other state-of-the-art results. Our method with 20,000 clusters of users achieves an RMSE of 0.799, which improves the result of CF-NADE by 0.004.

### 4.4 Running Time and Memory

We analyze the running time and memory of the proposed method. All the experiments are conducted on a single Nvidia TITAN X gpu with Theano [2] codes. As explained



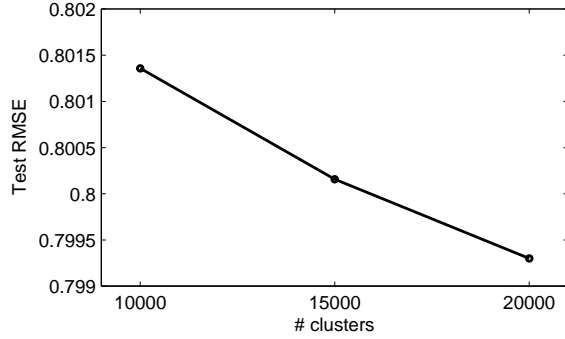


Figure 6: The performance on Netflix of CF-UIcA w.r.t. the number clusters of users.

Table 2: Test RMSE of various methods on Netflix. All the results are taken from [39].

Method	Test RMSE
LLORMA-Global (Rank-20)	0.874
U-RBM	0.845
BiasMF	0.844
LLORMA-Local (Rank-20)	0.834
I-AutoRec	0.823
U-CF-NADE-S (single layer)	0.804
U-CF-NADE-S (2 layers)	0.803
CF-UIcA (20K user clusters, $H^U = H^I = 500$ )	<b>0.799</b>

in Sec. 3.3, the minibatch sizes of users and items are not deterministic during training. Thus there is no standard way to train CF-UIcA epoch by epoch. We thus only report the time cost for each minibatch. Table 3 summarize the average running time of one minibatch by CF-UIcA on the two datasets.

The running memory cost by CF-UIcA is mainly for saving the 3-dimensional weight matrices. Specifically, the memory complexity of CF-UIcA is  $O((NH^U + MH^I)K)$ . In our experiments we always let  $H^U = H^I = H$ , resulting the memory cost is proportional to  $(N + M)HK$ . When dealing with large scale data, users and items can be clustered into  $N', M'$  groups, respectively. In this case, the memory cost is proportional to  $(N' + M')HK$ .

## 5. DISCUSSIONS AND CONCLUSIONS

In this paper, we propose CF-UIcA, which is a neural co-autoregressive model for collaborative filtering tasks. The model is co-autoregressive in the sense that it is autoregressive in the domains of both users and items, making it be able to capture both correlations between users and items explicitly and simultaneously. Under natural assumptions, we develop a stochastic learning algorithm to speed-up the training. A simple and effective method of “sharing param-

Table 3: Average running time for each minibatch of CF-UIcA on different datasets.

Dataset	$B_U/B_I$	$H^U/H^I$	Time
ML 1M	1,000/1,000	500/500	0.77s
Netflix	4,000/1,000	500/500	3.4s

eters in groups of users” is proposed to deal with large scale data. Experiments show that our method achieves state-of-the-art results, and is able to learn semantic information by visualization.

As analyzed above, the number of parameters of CF-UIcA is proportional to the sum of the number of users and the number of items. The proposed method of clustering users and items into groups can alleviate this problem to a certain extent. However, it breaks the end-to-end property of the overall method—The objective of the clustering procedure is different from the objective of CF-UIcA, which may lead to sub-optimal performance. For future work we like to investigate a more elegant way to deal with large scale data. Also, combining CF-UIcA with side information of users and items is another interesting challenge.

## 6. REFERENCES

- [1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
- [2] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.
- [3] J. Bennett and S. Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.
- [4] D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *ICML*, volume 98, pages 46–54, 1998.
- [5] L. Bottou. *Large-Scale Machine Learning with Stochastic Gradient Descent*. Physica-Verlag HD, 2010.
- [6] G. K. Dziugaite and D. M. Roy. Neural network matrix factorization. *arXiv preprint arXiv:1511.06443*, 2015.
- [7] M. D. Ekstrand, J. Riedl, and J. A. Konstan. Collaborative filtering recommender systems. *Foundations and Trends in Human-Computer Interaction*, 4(2):175–243, 2011.
- [8] P. Gopalan, J. M. Hofman, and D. M. Blei. Scalable recommendation with poisson factorization. *arXiv preprint arXiv:1311.1704*, 2013.
- [9] P. Gopalan, F. J. Ruiz, R. Ranganath, and D. M. Blei. Bayesian nonparametric poisson factorization for recommendation systems. *Artificial Intelligence and Statistics (AISTATS)*, 33:275–283, 2014.
- [10] P. K. Gopalan, L. Charlin, and D. Blei. Content-based recommendations with poisson factorization. In *Advances in Neural Information Processing Systems*, pages 3176–3184, 2014.
- [11] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.
- [12] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):19, 2015.
- [13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.



- [14] Y.-C. Juan, W.-S. Chin, Y. Zhuang, B.-W. Yuan, M.-Y. Yang, and C.-J. Lin. Libmf: A matrix-factorization library for recommender systems. <https://www.csie.ntu.edu.tw/~cjlin/libmf/>, 2016.
- [15] G. Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, pages 247–254, 2001.
- [16] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [17] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.
- [18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [19] H. Larochelle and S. Lauly. A neural autoregressive topic model. In *Advances in Neural Information Processing Systems*, pages 2708–2716, 2012.
- [20] H. Larochelle and I. Murray. The neural autoregressive distribution estimator. In *International Conference on Artificial Intelligence and Statistics*, pages 29–37, 2011.
- [21] V. D. M. Laurens and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2605):2579–2605, 2008.
- [22] N. D. Lawrence and R. Urtasun. Non-linear matrix factorization with gaussian processes. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 601–608. ACM, 2009.
- [23] J. Lee, S. Kim, G. Lebanon, and Y. Singer. Local low-rank matrix approximation. In *Proceedings of The 30th International Conference on Machine Learning*, pages 82–90, 2013.
- [24] A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2007.
- [25] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.
- [26] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994.
- [27] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, pages 880–887. ACM, 2008.
- [28] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007.
- [29] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, pages 285–295, 2001.
- [30] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Application of dimensionality reduction in recommender system – a case study. In *IN ACM WEBKDD WORKSHOP*, 2000.
- [31] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web Companion*, pages 111–112, 2015.
- [32] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
- [33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.
- [34] T. T. Truyen, D. Q. Phung, and S. Venkatesh. Ordinal boltzmann machines for collaborative filtering. In *Proceedings of the Twenty-fifth Conference on Uncertainty in Artificial Intelligence*, pages 548–556. AUAI Press, 2009.
- [35] B. Uria, I. Murray, and H. Larochelle. A deep and tractable density estimator. *JMLR*, 32(1):467–475, 2014.
- [36] B. Uria, I. Murray, and H. Larochelle. A deep and tractable density estimator. In *ICML*, pages 467–475, 2014.
- [37] A. van den Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*, pages 2643–2651, 2013.
- [38] M. Xu, J. Zhu, and B. Zhang. Nonparametric max-margin matrix factorization for collaborative prediction. In *Advances in Neural Information Processing Systems*, pages 64–72, 2012.
- [39] Y. Zheng, B. Tang, W. Ding, and H. Zhou. A neural autoregressive approach to collaborative filtering. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 764–773, 2016.